XML Types for C#

Wolfram Schulte
joint work with
Erik Meijer

Bill Gate wrote recently in a note on "A Software Driven Future" :

"*We need to come up with language extensions that make it easy to program in the XML world.*"

Why is that?

- XML is the lingua franca of the Internet
  - XML is typed content, which helps with interoperability.
- Reliability is crucial in the era of Web services.
  - Correctness, scaleability is a must
- But what is the current state of affairs?
  - A plethora of experimental, often untyped languages, without modul systems, libraries, versioning, ...

Motivation of Further Work

- New X… languages are restricted
  - Own syntax, new/no type system, no module system, limited library…
- Old C… languages are advanced
  - Standard syntax, type system, module system, rich library…

Why is that?

- XML is the lingua franca of the Internet
  - XML is typed content, which helps with interoperability.
- Reliability is crucial in the era of Web services.
  - Correctness, scaleability is a must
- But what is the current state of affairs?
  - A plethora of experimental, often untyped languages, without modul systems, libraries, versioning, ...

Languages and Approaches How to Use and Integrate XML

Language/ Feature	Purpose	Paradigm/ Syntax	Typesystem	Technology	Restrictions
XML Schema	Type Declarations	Data defs/ XML	XML	Validating Parser	Sublanguage
XPath	Projection	Functional/ Directory Paths	None	Interpreter/ Compiler	Sublanguage
XQuery	Query Language	Functional/ Own	XML	Prototype	Domain specific
XSLT	Transformation Language	Functional/ XML	None	Interpreter/ Compiler	Domain specific
XDuce	Explore DTD Typesystem	Functional/ SML	Monomorphic DTDs	Interpreter/ Compiler	Experimental
XMLambda	Explore DTD typesystem	Data defs; Fun/ Haskell + XML	Polymorphic DTDs	Not implemented	Experimental
System.XML	XML support for C#	Imperative/ C#	C#	Library	XML proc. untyped
XSD Compiler	XML support for C#	Data defs; Imp./ C#	C#	Compiler	C#-XML Type mismatch

Motivation of Further Work

- New X… languages are restricted
  - Own syntax, new/no type system, no module system, limited library…
- Old C… languages are advanced
  - Standard syntax, type system, module system, rich library…

The type and paradigm clash!

C languages (C#/Java/VB)

- Imperative, OO
- Simple Lexis
- Global types only
- Two kind of types
  - Value Type
  - Reference Type
- Element and singleton two distinct types
- Subtyping based on
  - Named relationships

X languages (XML/XPath/XQuery)

- Declarative
- Complex lexis
- Global and anonymous types
- Two kind of types
  - Simple types
  - Complex types
- Identification of element and singleton type
- Subtyping based on
  - Inclusion (simple types)
  - Structural Equivalence
  - Named relationships

Solution: XML Types for C#

The proposal is called:

C#-xml

The type and paradigm clash!

C languages (C#/Java/VB)

- Imperative, OO
- Simple Lexis
- Global types only
- Two kind of types
  - Value Type
  - Reference Type
- Element and singleton two distinct types
- Subtyping based on
  - Named relationships

X languages (XML/XPath/XQuery)

- Declarative
- Complex lexis
- Global and anonymous types
- Two kind of types
  - Simple types
  - Complex types
- Identification of element and singleton type
- Subtyping based on
  - Inclusion (simple types)
  - Structural Equivalence
  - Named relationships

XML Types and Values are First Class Citizens in C#-xml

- XML types can be denoted
- XML values can be constructed, loaded, passed, transformed, updated, and written in a type safe manner
- This improves reliability and performance

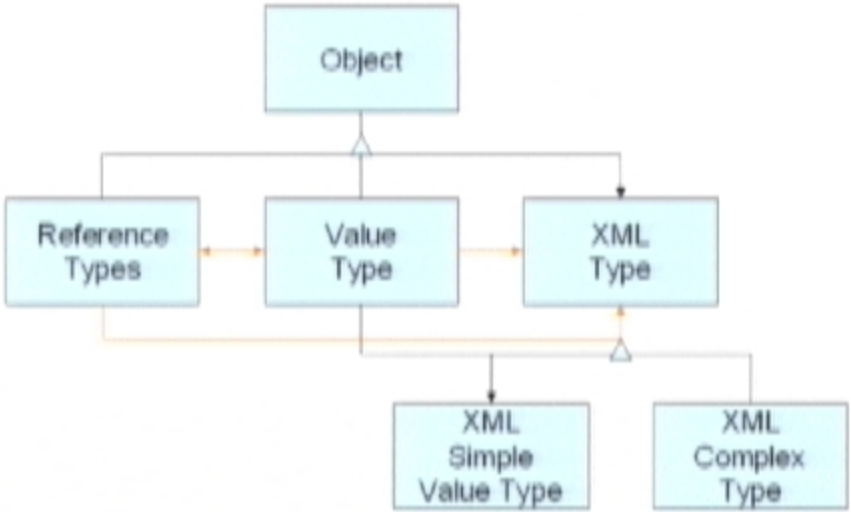
Design: Types

Add XML schemas as C# types

- C# value types stand in for XML *simple types*; implicit conversions replace inclusion
- *Complex types* are new; structural subtyping supported; (named subtyping not yet)
- New *sequence type* and iterators are added

- Type system from [XML Schema: Formal Description]
- Data model from [XQuery 1.0 and XPath2 Data Model]

The Combined Type System



(simplified)

Example Bib Schema

```
<?xml version = "1.0" encoding = "UTF-8"?>
<!-- Conforms to w3c http://www.w3.org/2001/XMLSchema-->
<xsd: schema xmlns:xsd = "http://www.w3.org/2001/XMLSchema">
    <xsd: element name = "bib" type = "Bib"/>
    <xsd: element name = "book" type = "Book"/>

    <xsd: complexType name = "Bib">
        <xsd: sequence>
          <xsd: element ref = "book" minOccurs = "0" maxOccurs = "unbounded"/>
        </xsd:sequence>
    </xsd: complexType>

    <xsd: complexType name = "Book">
        <xsd: sequence>
            <xsd: element name = "title" type = "xsd:string"/>
            <xsd: element name = "author" type = "xsd:string" maxOccurs = "unbounded"/>
        </xsd:sequence>

        <xsd: attribute name = "price" use = "optional" type = "xsd:float"/>
        <xsd: attribute name = "isbn" use = "required" type = " xsd:string "/>
    </xsd: complexType
</xsd: schema>
```

Type Representation

New typeconstructors

group, **elem**, **attr**, **choice**, **all**. *T*[*min*-*max*]

The Bib schema

```
group Bib{
  elem bib {Book}[0-*];
}
group Book {
  attr price {float} [0-1];
  attr isbn {string};
  elem title {string};
  elem author {string [1-*]};
}
```

Note. These types are rarely used – use XSD instead!.

XML Subtyping (<:)

- Simple Types:
  - Inclusion of value spaces
  - normalizedString <: string

- Complex Types
  - Inclusion of sets of sequences they denote
  - Let t1, t2 be types, then

    t1 <: choice{ t1:t2 } and t2 <: choice{ t1:t2}
  - Let t, t' be types, then if t <: t' and m' <= m and n <= n' then

    t[m-n] <: t'[m'-n'].

Example Bib Document

```
<?xml version = "1.0" encoding = "UTF-8"?>

    <bib xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
      xsi:noNamespaceSchemaLocation = "file:///C:/XML/XML/Bib.xsd">

    <book isbn = "0-201-70914-7">
          <title> Essential XML </title>
          <author> Box </author>
          <author> Skonnard </author>
          <author> Lam </author>
    </book>

    <book isbn = "0-201-17888-5">
          <title> Component Software </title>
          <author> Szyperski </author>
    </book>
</bib>
```

XML Namespaces and Literals

```
using System;
using Bib.xsd;                                   ←Namespace

public class Sample {

  private static Book m =
   <book isbn = "0-201-17888-5">                 ← Literal
          <title>Component Software</title>
          <author>Szyperski</author>
   </book> ;

  public static void Main() {
   Console.WriteLine(m);
}}
```

Multiple and Default Namespaces

```
using ph = "http://www.PrenticeHall.com/books.xsd";      ←Namespaces
using aw = "http://www.AddisonWesley.com/books.xsd";

using default springer = "http://www.Springer.com/books.xsd";

public class Sample {
                                                          ← Lexis Quotes
    static "aw:Book" b1 = <aw:book ... > ... </aw:book>;
    static "ph:Book" b2 = <pw:book ... > ... </pw:book>;
    static Book b3 = <book ... > ... <book>;

  ...
}
```

XML Quotes

```
private static Book createBook (int year, string isbn, string title, string[]
    authors) {
 return
   <book isbn={ isbn }>                                    ← Quote expression
     <title>{ title } </title>
     {                                                     ← Quote statement
     for (int i = 0 ; i < authors.Length; i++) {
       yield <author>{ authors[i] }</author>;              ← Quote result
     }
   </book>;
  }

  public static void Main() {
   Console.WriteLine(createBook(1999, "0-201-17888-5",
          "Component Software", new string [] {"Szyperski"}));
}
```

Projection

New Expression:

get *xpath*

Select sequence of nodes: here find all author elements

get bib/book/author

It typechecks as follows:

bib	has type	Bib
bib/book	has type	Book[0-*]
bib/book/author	has type	{elem author { string } } [0-*]

More Projections

Find all book elements which appeared in 2000:

```
Book[*] q2 = get bib/[ /@year == 2000] ;
```

predicates

Find the last author string in the bib data base:

```
String q7 = Seq.Last( get bib/book/author/data() );
```

aggregation

Find all author elements, sorted by names without duplicates

```
Author[*] q7 = Seq.DistinctValue( get bib/book/author sortby /data());
```

Updates

New Statements

```
set xpath = expr;
```

Select the node (sequence) whose content is updated;
here update a book attribute

```
set book/@price = 25.0;
```

C#-xml may introduce runtime check to guarantee type correctness. Same as the array update problem in C#.

More Updates

- *Update the title of a book element:*
  set book/title = ← reuse node
  <Title>{(get value /data()).ToUpper() }</Title>;

- *Update all authors to uppercase*
  set bib/book/author = ← multiple updates
  <author>{ (get value/data()).ToUpper() }</author>

- *Add book b to bib database before the author Box*
  set before bib/book[author/data()="Box] = b; ← before and after

- *Delete all books where Box is one of the authors*
  delete bib/book[./@author == "Box"]; ← deletion

Updates: Parameter passing

XML values are math values, therefore

- Only (parts of) variables containing XML values can be updated.
- For updates parameter must be passed by reference

Increase price by ten percent

```
static inc (ref Book b) {
 set b/@price = 25
      }
```

Iteration, Join

Extended **foreach** statement

foreach (var **in** *projection*, … , var **in** *projection*) {…}

Select from two databases all books that have overlapping authors but different titles

```
Book [0-*] res;
foreach (Book b1 in get bib1/book,
         Book b2 in get bib2/book,
         String a1 in get b1/author/data(),
         String a2 in get b2/author/data()
         [a1 == a2 && b1/title/data() != b2/title/data()] )
    res +=b1; res +=b2; }
```

Input/Output

Extended typed IO:

```
using Bib.xsd;
using System.XML;

public static void Main(string [] args) {
   readonly Bib bibIn = Bib.OpenRead(arg[0]);
   writeonly Bib bibOut = Bib.OpenWrite(arg[1]);

   bibOut = <bib> {StripYearAndPrice(get bibIn/Book)} </bib>;

   bibIn.Close();
   bibOut.Close();
}}
```

Stream-Iterators

Iterators for processing information piece by piece:

```
static iterator Book[0-*] StripPrice (Book[0-*] is) {
  foreach (Book b in is)
    yield <book {b/@isbn}> {b/title}{b/author} </book>;
}
```

Summary

X# extends C# by

- Full support of XSD
  - But XML values live in separate value space
- More than XQuery
  - But imperative
- A little bit for Streamprocessing
  - But can hopefully be extended

Thanks for coming!

Most recent version can be found at
http://msrweb/fse/schulte/XMLIntegration.doc

Input/Output

Extended typed IO:

```
using Bib.xsd;
using System.XML;

public static void Main(string [] args) {
   readonly Bib bibIn = Bib.OpenRead(arg[0]);
   writeonly Bib bibOut = Bib.OpenWrite(arg[1]);

   bibOut = <bib> {StripYearAndPrice(get bibIn/Book)} </bib>;

   bibIn.Close();
   bibOut.Close();
}}
```

Thanks for coming!

Most recent version can be found at
http://msrweb/fse/schulte/XMLIntegration.doc